VISIT…

Elektrotehniški vestnik 65(5): 267-271, 1998
Electrotechnical Review, Ljubljana, Slovenija

SPICE for Windows 95/98/NT

Janez Puhan, Tadej Tuma, Iztok Fajfar

Univerza v Ljubljani, Fakulteta za elektrotehniko,
Tržaška cesta 25, 1000 Ljubljana, Slovenija
E-pošta: janez.puhan@fe.uni-lj.si

Abstract. The analogue circuit simulator SPICE was originally developed at the University of California at Berkeley. It was has been constantly upgraded and maintained at Berkeley from 1972 to 1992, when the last official release 3f4 was published. The UNIX based source code is still available as freeware at *ftp://ic.berkeley.edu/pub/Spice3/*. Although the research at Berkeley has stopped, many commercial companies continue to use the Berkeley code as the core of their SPICE compilations with no or little modification. Since we have good experience with the original SPICE 3f4 code on UNIX machines, we decided to modify the code in order to run it on the PC running Windows operating system. In this paper, we present a fully functional compilation of SPICE 3f4 for Windows 95/98/NT. Our compilation offers a native SPICE environment with an interactive interpreter known from UNIX operating systems. In the paper, a PC version is introduced through two simulation cases. All simulation files as well as the simulator itself may be downloaded from *http://fides.fe.uni-lj.si/spice*.

Key words: SPICE, CAD, Nutmeg, analog circuit analysis

SPICE za Windows 95/98/NT

Povzetek. Program SPICE je namenjen analizi analognih vezij. Razvit je bil na Kalifornijski univerzi Berkeley v letih od 1972 do 1992, ko je bila objavljena zadnja uradna različica 3f4, razvoj programa pa ustavljen. Originalna izvorna koda pisana v UNIX okolju je javna last in je še vedno na voljo na naslovu *ftp://ic.berkeley.edu/pub/Spice3/*. Berkeleyevo izvorno kodo brez ali z le manjšimi spremembami uporabljajo v svojih izdelkih mnogi komercialni ponudniki SPICE-a.
Po dobrih izkušnjah z izvorno kodo v UNIX okolju smo se odločili, da jo prilagodimo za tek na osebnih računalnikih z Windows operacijskimi sistemi. V članku predstavljamo SPICE 3f4 za Windows 95/98/NT v popolni obliki. Naša različica ponuja prvotno SPICE okolje z interaktivnim interpreterskim jezikom, ki ga poznamo z UNIX operacijskih sistemov. Nova različica je v članku predstavljena z dvema primeroma. Vse datoteke, kot tudi program, je možno dobiti na naslovu *http://fides.fe.uni-lj.si/spice/*.

Ključne besede: SPICE, CAD, Nutmeg, analiza analognih vezij

1 Introduction

SPICE (Simulation Program with Integrated Circuit Emphasis) is the most commonly used analogue circuit simulator today. Through years it has become a nonofficial industrial standard for computer aided design of electronic circuits. SPICE is a general-purpose analogue simulator. It contains models for most circuit elements and can handle large non-linear circuits. The simulator can perform several different types of circuit analyses. The most important are the DC analysis, AC small-signal analysis and transient analysis, which is numerically the most complex analysis in SPICE.
The simulator was developed at the University of California, Berkeley, and was first released in 1972. Many scientists at Berkeley and other institutions have contributed to the development and improvement in subsequent versions of SPICE. The next major release of SPICE, called SPICE2, was published in 1975 [1]. The core of the program has remained intact, even after many improvements and additions. The last major release, SPICE3 [2], came in 1985 with a conversion of the source code from FORTRAN to the C programming language. The source code of Berkeley's SPICE is public domain. In the last officially published version 3f4, the program is divided into two parts: the simulator and the front-end. The front-end includes an interactive interpreter programming language, which allows interactive SPICE sessions. It acts as a simple pre- and post-processor. The

Received 8 October 1998
Accepted 23 October 1998

source code can be compiled on different hardware platforms with different operating systems. To a limited extent it can also be compiled on a IBM compatible personal computer with a MS-DOS operating system.
Many companies like Intusoft (*http://ww.intusoft.com*) have compiled SPICE and integrated it as an analogue circuit analyser into many different development environments including programs for design of printed circuit boards, digital circuit simulation, filter design tools, mixed mode simulators etc. Their efforts are mainly focused into building user-friendly graphical pre- and post-processors and building libraries of models. These commercial SPICE compilations are usually very expensive, although their numerical performance is rarely superior to the original Berkeley public domain. For these reasons we decided to compile Berkeley's latest SPICE version for PCs running Windows 95/98/NT and, of course, make it public domain.

2 Compiling SPICE for MS-DOS and Windows

Our starting point was the Berkeley's SPICE 3f4 code from *http://www.cad.eecs.berkeley.edu/Software/software.html*. It can be directly compiled for the MS-DOS operating system on IBM compatible personal computers with a slightly out-of-date Microsoft C compiler version 5.1. The compilation produces the following executables:

- *bspice* a batch mode simulator (the simulator part of SPICE)
- *cspice* a SPICE2 like interface for small runs
- *nutmeg* a standalone data analysis program (the front-end part of the SPICE without the help command and any graphical interface)
- *help* a standalone help browser (the help command of the front-end part)
- *proc2mod* converts process characterisation files to *BSIM1* MOS model definitions
- *sconvert* converts between ascii and binary SPICE data files
- *multidec* a utility for decomposing coupled lossy transmission lines into equivalent uncoupled lines

All these executables are only a small part of the entire integrated simulator environment and therefore not so useful for serious work. There isn't even a graphical postprocessor. Hence our motivation to compile the main executable *spice3* under Windows. It allows interactive SPICE sessions, programming capabilities, and graphical data presentation. All attempts to compile *spice3* with the Microsoft C 5.1 compiler failed because of memory problems. So we tried with the latest Microsoft Visual C++ 6.0 compiler. After solving some incompatibility problems we managed to compile and run *spice3* in an MS-DOS window under Windows 95/98/NT with two serious drawbacks:

1. As we changed the compiler as well as the operating system, all graphical routines are incompatible. So we lost all graphical support. *Spice3* was running without its *plot* and *iplot* [3] commands.
2. The resulting code was extremely unstable. It was not convenient for longer interactive SPICE sessions because of numerous memory leaks of the original code. They were usually fatal and caused the program to crash. The problem turned out to be sloppy programming all over the *nutmeg* portion of the code. There are also many attempts of dereferencing NULL pointers. The Windows operating system is very strict about this while UNIX proves to be quite tolerant towards dereferencing uninitialized pointers.

We overcame the first problem by rewriting all graphical functions using Windows graphical libraries. The original syntax of the *plot* and *iplot* commands was of cause completely preserved, enabling any script compatibility with earlier SPICE 3f4 source files. The second problem took many months of painful searching for needles in the haystack, but finally we were able to produce quite a stable PC version of SPICE 3f4.

3 An advanced simulation case

Let us consider the audio power amplifier in Figure 1. The relatively strong negative feedback of the resistors *R3* and *R4* is defining a constant gain of approximately 100. At the same time the negative feedback is causing a linear response and is also keeping the output impedance low. In our case, we will analyse the output impedance in detail. The two complementary power transistors *Q7* and *Q8* are taking care of the positive and negative output current respectively, which is drawn by the amplifier's load resistor *R13*. The process of one transistor passing the load control to the other is always a source of non-linear behaviour, which is usually reduced by some voltage difference between the two base terminals of the complementary transistors. In our case, transistor *Q4* in conjunction with the resistors *R9* and *R10* is smoothening the crossover. In addition to this mechanism, the negative feedback is also compensating for this disturbance. Therefore it should be possible to show that the negative feedback impact on the output resistance is reduced at very low output voltages, since this is the area where the feedback must compensate for the crossover in the bias.

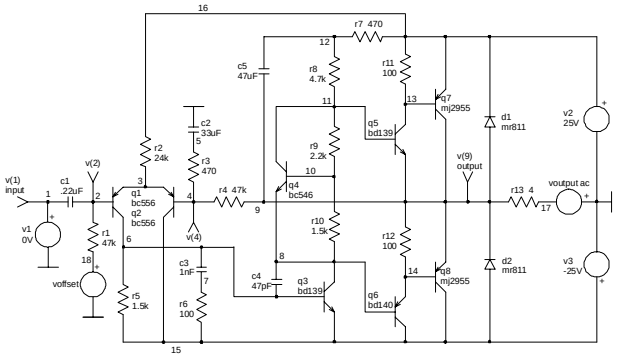

Figure 1: An audio power amplifier

Let us first check the situation at the base terminals of the two Darlington transistors. We will run a *dc* analysis sweeping the input offset voltage source from –200mV to 200mV, while observing the voltage at the base nodes 8, 11, and the common emitter node 9. After having loaded the circuit description file *poweramp.cir* it takes only two additional commands to obtain a plot window with appropriate results (Figure 2).

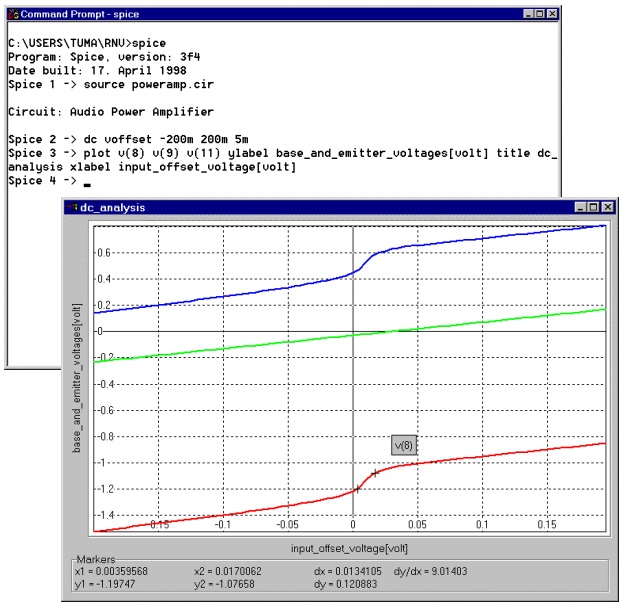

Figure 2: The crossover compensation

As can be clearly seen from the markers in Figure 2, we have a considerable step in both base voltages, which is caused by the negative feedback in order to produce a linear output voltage (the curve in the middle). Let us now look at the effect of the crossover on the output impedance, which can be determined as the quotient between the small signal response of the output voltage *v(9)* versus the output current *i(voutput)* over a certain frequency range. The circuit bias point is, of course, a very important parameter in the *ac* analysis. We will run five ac analyses at offset voltages from 0mV to 40mV by simply typing the following commands shown in Figure 3.

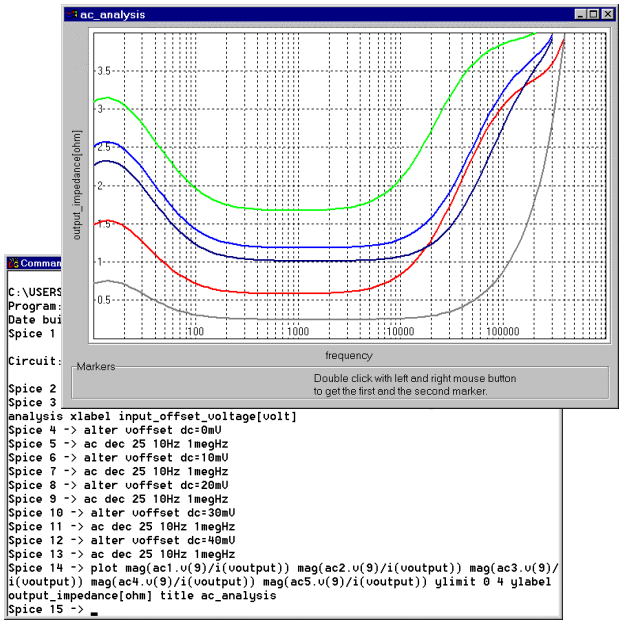

Figure 3: The output impedance at different biases

The resulting curves yield a typical *ac* response of the output impedance. The flat sections clearly identify the amplifiers active frequency range, where the negative feedback is reducing the output impedance. Also, the effect of the offset voltage in the crossover range is apparent. Let us investigate this effect further by running a multiple *ac* analysis in order to plot the output impedance versus the offset voltage at the central frequency of 1KHz. Although we could continue our dialog from Figure 3, these analyses seem to be too complex to type interactively, therefore we decided to write a simple script file containing the appropriate commands in a *while* loop. The script might look like the one depicted in Figure 4.

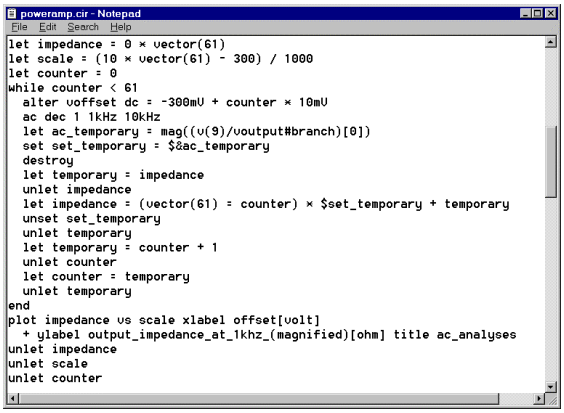

Figure 4: The output impedance vs. the input offset

The *plot* command after the *while* loop summons the simulation data and displays the graphical results shown in Figure 5.

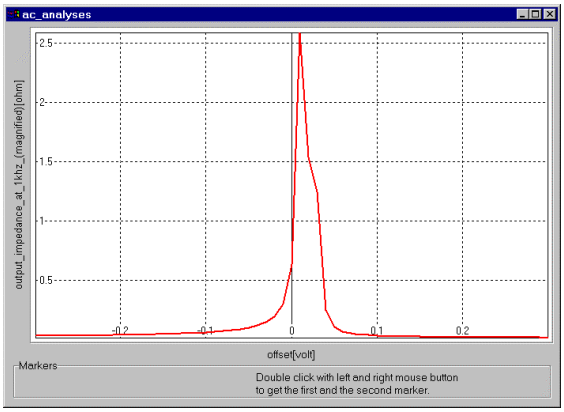

Figure 5: The output impedance vs. the input offset

As expected, we can see a dramatic increase in the output impedance during the crossover. This is a very interesting analysis, but of no particular practical value since the effect is limited to a range of output voltages from 0 to 50mV. For audio purposes this range is certainly neglectable. The example nevertheless shows how easy and fast even very complex analyses can be done using the original Berkeley compilation of SPICE 3f4 on a Windows95 platform. In our next example we will show that it is even possible to do optimisation loops directly within the *nutmeg* user interface.

4 An optimisation example

We are looking for values of the two resistors *R1* and *R2* in a simple amplifier in Figure 6. We want maximal linearity at an amplification factor of at least 20000V/A.

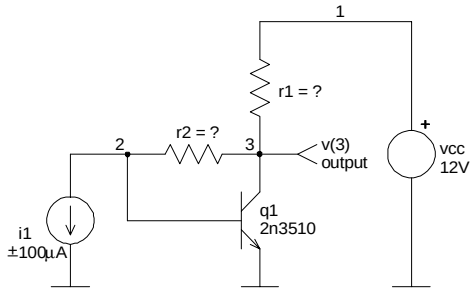

Figure 6: A simple transistor amplifier

First we have to define some explicit as well as one implicit constraint. The explicit constraints are obvious since we know that the resistor values can't possibly lie outside the intervals:

$$5\text{k}\Omega \le R_1 \le 50\text{k}\Omega \qquad (1)$$
$$20\text{k}\Omega \le R_2 \le 200\text{k}\Omega$$

Since we intend to analyse the circuit in the *dc* domain, we define the amplification as the ratio of the extreme points of the response to the excitation. This is our implicit constraint:

$$A = \frac{v_3(I_{\max}) - v_3(-I_{\max})}{2I_{\max}} \ge 20000\text{V/A} \qquad (2)$$

The most important part of any optimisation is certainly the cost function. In our case we want the cost function to reflect the circuit linearity, defined as a normalised square area between the real and ideal response:

$$E(R_1,R_2) = \frac{\int_{-I_{\max}}^{I_{\max}} (Ai_1 - (v_3(i_1) - v_3(0)))^2 \, di_1}{\int_{-I_{\max}}^{I_{\max}} (Ai_1)^2 \, di_1} \qquad (3)$$

The cost function must be a positive scalar and will be minimised during the optimisation procedure. The ideal value is zero, in which case the amplifier *dc* response is linear.

The implementation of the implicit constrain and the cost function into the *nutmeg* source code is fairly simple:

```
**** The analysis
alias analysis dc i1 -100uA 100uA 10uA
**** The cost function definition
alias cost let temporary = (3 *
mean((((v(3)[20] -
  + v(3)[0]) / (2*100uA)) * sweep - v(3) +
v(3)[10])^2))
  + / (((v(3)[20] - v(3)[0]) / (2 *
100uA))^2 *
  + (100uA)^2)
**** The implicit constrain definition
alias gain let a = (v(3)[20] - v(3)[0]) /
(2 * 100uA)
```

As for the optimisation algorithm, we decided to use a well known constrained simplex optimisation method introduced in 1965 by M. J. Box [4]. It is robust and simple. Since the implementation of the optimisation algorithm takes over 160 *nutmeg* programming lines, we will not discuss it any further at this point. The entire simulation file *simplex.cir* can be downloaded from *http://fides.fe.uni-lj.si/spice.*

The execution of our optimisation loop takes *104* iterations and stops at the implicit constrain with a cost function value of *0.0009*, gain value of *20000.05*V/A and resistor values of *R1* = 25965Ω and *R2* = 33816Ω respectively. The optimisation takes about 17 seconds on a PENTIUM 133MHz personal computer running Windows NT.

5 Conclusions

We presented a fully functional compilation of Berkeley's latest SPICE 3f4 for the Windows platforms. Our compilation is free of charge and can be downloaded from *http://fides.fe.uni-lj.si/spice.* The code is strictly Berkeley compatible, thus directly running any circuit and library file from Berkeley. Also, any original documentation applies to our compilation.

Since we used the Microsoft's visual C compiler version 6.0, the resulting executable is relatively fast and will be easy to upgrade to new Windows releases in future. Although the graphical interface was completely rewritten in order to comply with the current Windows standards, its input syntax was not changed.

In this paper, we demonstrated that the front-end part of SPICE 3F4 (nutmeg) is very convenient both for interactive user sessions as well as for script file processing. The first simulation case shows the possibilities of a mixed interactive and script approach, combining the results of several different analyses into complex plots. In the second example, we pushed the script processing capabilities to the extreme by running a complete two-dimensional optimisation loop on a simple circuit.

The optimisation possibilities are actually our next goal. We intend to add an optimisation command to the NUTMEG interpreter, executing different constrained optimisation algorithms directly in the machine code. This should enable a relatively fast optimisation of large circuits [5].

6 References

[1] Laurence W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits", Memorandum No. ERL-M520, University of California, Berkeley, 1975

[2] Thomas L. Quarles, "Analysis of Performance and Convergence Issues for Circuit Simulation", Memorandum No. ERL M89/42, University of California, Berkeley, 1989

[3] T. Quarles, A. R: Newton, D. O. Pederson, A. Sangiovanni - Vincentelli, "SPICE3 Version 3f3 User's Manual", Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, 1993

[4] M. J. Box, "A new method of constrained optimization and comparison with other methods", Computer Journal, volume 7, pages 42 - 52, 1965

[5] J. Puhan, T. Tuma, "Optimization of analog circuits with SPICE 3f4", Proceedings of the ECCTD'97, Volume 1, pages 177 - 180, 1997

Janez Puhan graduated from the Faculty of Electrical Engineering, University of Ljubljana, Slovenia in 1993, where he also obtained his M.Sc. degree in 1998. His current research interests are in cumputer aided design of analog circuits, optimisation methods and computer circuit analysis.

Tadej Tuma received his diploma degree at the Faculty of Electrical Engineering in Ljubljana in 1988. Soon thereafter he joined the Faculty as a teaching assistant. At the same time he began his postgraduate studies, which were completed by his M.Sc. thesis in 1991 and his Ph.D. dissertation in 1995. His research interest is mainly in the field of computer aided circuit design, especially in analog circuit optimization methods.

Iztok Fajfar received the Dipl. Ing., M.Sc. and Ph.D. degrees in electrical engineering from the University of Ljubljana in 1991, 1994 and 1997, respectively. Since 1992 he has been with the Faculty of Electrical Engineering of the University of Ljubljana where he is currently an assistant professor. His research interests include design and optimisation of electronic circuits with focus on cellular neural networks.